结合项目流行度加权的协同过滤推荐算法

魏甜甜，陈　莉†，范婷婷，吴小华

(西北大学 信息科学与技术学院，西安 710127)

摘　要：针对传统协同过滤算法中存在的流行度偏差问题，提出一种结合项目流行度加权的协同过滤推荐算法。在项目协同过滤算法的基础上，分析项目流行度和流行度差异对相似度的影响；设置流行度阈值，对大于该阈值的流行项目设计惩罚权重，降低其对项目间相似度的贡献。通过在 MovieLens1M 和 Epinion 数据集上进行实验验证和对比，结果表明，所提算法的预测准确度和覆盖率均优于传统算法，有效提高了推荐的多样性和新颖性，一定程度上缓解了流行度偏差问题。

关键词：协同过滤；相似性度量；流行度偏差；项目流行度

中图分类号：TP391　　doi: 10.3969/j.issn.1001-3695.2018.08.0618

Collaborative filtering recommendation algorithm based on item popularity weighting

Wei Tiantian, Chen Li†, Fan Tingting, Wu Xiaohua

(*School of Information Science & Technology*, *Northwest University*, *Xi'an* 710127, *China*)

Abstract: Aiming at the popularity bias problem in traditional collaborative filtering algorithms, this paper proposes a collaborative filtering recommendation algorithm combined with item popularity weighting. On the basis of the item collaborative filtering algorithm, it analyzes the influence of item popularity and popularity difference between items on similarity. The algorithm, by analyzing the influence of item popularity and popularity difference on item similarity, uses the item popularity and popularity difference designed the penalty weight functions to adjust the similarity between popularity items when the item popularity is greater than the threshold. The experiments on the MovieLens1M and Epinion datasets show that the proposed algorithm has better prediction accuracy and coverage than traditional algorithms, which effectively improves the diversity and novelty of recommendations, and alleviates the popularity bias problem.

Key words: collaborative filtering; similarity measure; popularity bias; item popularity

0　引言

互联网技术的迅猛发展将人们带进了数据爆炸的时代，海量数据的同时呈现，加剧了信息过载问题[1]。推荐系统作为缓解信息过载的有效手段之一，已经广泛地被应用到电子商务(Amazon、京东)、电影和视频(Youtubo、MovieLens)等领域并取得较大进展。

协同过滤 (collaborative filtering, CF) [2]是目前应用最广泛、最成功的推荐算法，它的基本假设是：如果用户对一些项目的评分比较相似，则他们具有相似的偏好，因此他们对其他项目的评分也相似。CF 主要分为基于用户的协同过滤算法(user based collaborative filtering, UBCF)和基于项目的协同过滤算法(item based collaborative filtering, IBCF)。UBCF 通过计算用户间的相似度得到“邻居”用户集合，帮助目标用户找到可能会感兴趣的项目；而 IBCF 则通过计算项目间的相似度得到“邻居”项目集合，根据目标用户的历史偏好记录，为其推荐相似的项目。

为了找到“邻居”集合，一般要度量用户或项目间的相似度，不同的相似性度量函数，产生不同的“邻居”集合，最终影响推荐结果。因此利用合适的相似性度量函数产生偏好相似的“邻居”是整个协同过滤推荐算法的关键[3]。在实际的系统中存在数据稀疏性问题[4]，传统的相似性度量方法[5, 6]加剧了数据的稀疏性，导致部分相似度计算欠准确。针对该问题，Ahn[7]提出一种新的用户相似性度量方式 PIP(proximity-impact-popularity)，利用三种因子共同计算用户间的相似度，推荐质量优于传统的相似性度量方法。Ekstrand 等人[8]在 RecSys 年会上公布了开源工具 LensKit，提出了一种均值标准化的余弦相似度计算方法。在计算两个项目的相似度时，考虑到只对其中一个项目有评分的数据惩罚项目间的相关性，缓解了相似度计算欠准确的问题。上述两种方法[7, 8]通过改进相似度计算公式缓解数据稀疏性问题，以达到提高推荐准确度的目的，但是并没有考虑到流行度偏差现象[9~11]，导致推荐结果的覆盖率较低。针对该问题，Zhao 等人[12]利用项目流行度阈值设计权重函数，降低流行项目对用户相似度的影响。实验结果表明，该方法一定程度上缓解了 UBCF 存在的流行度偏差现象，且推荐准确度优于传统的 UBCF。Chen 等人[13]通过分析流行度对推荐结果的影响，提出利用可调节的项目流行度改进基于用户偏好的方法，实验结果证明适度地项目流行度调节可以提高推荐的准确性和多样性。但是随着用户和项目的不断加入，改进的 UBCF 可能会面临可扩展性问题。王锦坤等人[14]在 IBCF 的基础上，提出一种考虑用户活跃度和项目流行度的协同过滤算法，根据流行度差异阈值对项目相似度设计惩罚权重，一定程度上缓解了可扩展性问题，提高了数据稀疏环境下流行度较低的项

收稿日期：2018-08-14；**修回日期**：2018-10-09

作者简介：魏甜甜（1993-），女，陕西咸阳人，硕士研究生，主要研究方向为个性化推荐；陈莉（1963-），女（通信作者），陕西西安人，教授，博导，博士，主要研究方向为智能信息处理、数据挖掘、网络安全（chenli@nwu.edu.cn）；范婷婷（1990-），女，山东泰安人，硕士研究生，主要研究方向为协同过滤；吴小华（1993-），男，陕西安康人，硕士研究生，主要研究方向为自然语言处理.

目被推荐的概率。但是该方法仅仅对部分项目设计惩罚函数，忽略了同时对两个项目有评分的数据，相似度计算欠准确。

结合项目流行度改进的 IBCF 对于缓解流行度偏差现象取得了一定的效果，但是在计算项目间的相似度时，忽略了两个项目由于流行度不同，对相似度会产生不同的影响。流行度高的项目被用户同时选择和评价的可能性较大，导致流行项目间的相似度普遍偏高。针对该问题，本文设置流行度阈值，对大于该流行度阈值的项目，利用项目流行度和流行度差异分别设计权重函数，并将其引入到项目相似度计算过程中，实现了项目流行度加权的相似性度量方法 popularity weighting-IBCF(PW-IBCF)。本文在 MovieLens1M 和 Epinion 数据集上进行实验仿真，实验结果表明，本文算法能够有效改善流行度偏差问题，提高推荐的多样性和新颖性。

1　相关工作

1.1　基于项目的协同过滤算法

基于项目的协同过滤基本思想是计算项目间的相似度，选取目标项目的“邻居”项目集合，将类似的物品推荐给目标用户。假设 $U=\{u_1,u_2,...,u_m\}$ 表示 m 个用户集合，$I=\{i_1,i_2,...,i_n\}$ 表示 n 个项目集合，用户对项目的评分信息使用 m×n 阶的用户-项目评分矩阵 R 表示，$R=[r_{ui}]^{m\times n}$。表 1 给出了用户项目评分矩阵的例子。其中，r_{ui} 表示用户 u 对项目 i 的评分；$r_{ui}=0$ 代表该用户对该项目未产生评分。

表 1　用户—项目评分矩阵

Table 1　Example of user-item rating matrix

r_{ui}	i_1	i_2	i_3	i_4
u_1	2	3	0	0
u_2	1	0	0	5
u_3	0	0	0	4
u_4	0	0	0	0

得到用户项目评分矩阵后，基于项目的协同过滤算法如下：

a)计算目标项目 i 与系统中其他项目 $j\in I$ 的相似度 $sim(i,j)$。利用项目相似度计算公式计算项目间的相似度，得到项目相似度矩阵 $M_{item}(i,j)$。

b)确定目标项目的近邻集合 N_i。根据 $M_{item}(i,j)$ 为目标项目 i 选取相似度最大的 K 个项目作为近邻项目集合 N_i。

c)产生评分预测。根据目标项目 i 与近邻项目 $j\in N_i$ 的相似度 $sim(i,j)$ 和评分值 r_{uj} 计算评分预测值 P_{ui}，预测方法如式(1)所示。

$$P_{ui}=\frac{\sum_{j\in N_i} sim(i,j)r_{uj}}{\sum_{j\in N_i} sim(i,j)} \tag{1}$$

1.2　传统的相似性度量方法

传统的项目相似度计算公式[5]主要有三种，包括余弦相似度、修正的余弦相似度和皮尔森相关系数。

余弦相似度：项目评分被看成是 m 维用户空间上的向量，项目间的相似性通过向量间的余弦夹角度量。设项目 i 的和 j 的评分向量为 $\vec{r_i}$ 和 $\vec{r_j}$，余弦相似度计算方法如式(2)所示。

$$sim(i,j)^{COS}=\frac{\vec{r_i}\cdot\vec{r_j}}{\|\vec{r_i}\|\times\|\vec{r_j}\|} \tag{2}$$

修正的余弦相似度：余弦相似度没有考虑不同用户的评分尺度问题，修正的余弦相似度通过减去不同用户的评分均值来改善上述问题。假设对项目 i 和 j 同时评分过的用户集合为 $U_i\cap U_j$，$\bar{r_u}$ 表示用户 u 的评分均值，修正的余弦相似度计算方法如式(3)所示。

皮尔森相关系数：反映两个变量线性相关程度，也被广泛用于协同过滤相似性的度量。假设 $\bar{r_i}$ 和 $\bar{r_j}$ 分别表示项目 i 和 j 的评分均值，皮尔森相关系数计算方法如式(4)所示。

$$sim(i,j)^{ACOS}=\frac{\sum_{u\in U_i\cap U_j}(r_{ui}-\bar{r_u})(r_{uj}-\bar{r_u})}{\sqrt{\sum_{u\in U_i\cap U_j}(r_{ui}-\bar{r_u})^2}\sqrt{\sum_{u\in U_i\cap U_j}(r_{ui}-\bar{r_u})^2}} \tag{3}$$

$$sim(i,j)^{PCC}=\frac{\sum_{u\in U_i\cap U_j}(r_{ui}-\bar{r_i})(r_{uj}-\bar{r_j})}{\sqrt{\sum_{u\in U_i\cap U_j}(r_{ui}-\bar{r_i})^2}\sqrt{\sum_{u\in U_i\cap U_j}(r_{ui}-\bar{r_j})^2}} \tag{4}$$

1.3　流行度偏差现象

流行度偏差[12]是传统协同过滤算法中普遍存在的一种现象，流行度越高的项目被推荐得越频繁，而流行度较低的项目则不容易得到推荐。统计 MovieLens1M 用户评分数据集的项目流行度，其中流行度较低(小于项目流行度均值)的项目约占 70%，流行度较高的项目约占 30%，流行度较低的项目明显比流度高的项目多。然而传统的协同过滤算法产生的推荐列表中流行项目较多，导致多样性和新颖性较低。因此，在推荐系统中需要减少流行项目被推荐的机会以缓解流行度偏差现象。

2　结合项目流行度加权的协同过滤推荐算法

传统的项目协同过滤算法在计算相似度时忽略了项目流行度的影响，导致流行度低的项目不容易得到推荐。为了有效地缓解流行度偏差现象，需要对流行度较高的设计相似度惩罚函数。通过分析项目流行度和流行度差异对项目相似度的影响，对流行度较高的利用项目流行度与项目流行度差异对相似度设计惩罚权重，降低流行项目间的相似度以缓解传统相似性度量方法存在的不足，使得项目间的相似度比较准确。

2.1　项目流行度

项目流行度一般是指该项目被用户评分的用户数[13]，某个项目被评价的次数越多，该项目的流行度越高。流行度高的项目可能由于其知名度或者性价比高被大多数用户评价，如果存在共同评分项目，并不代表该流行项目与其他项目相似。对于每一个项目 i，统计对其评分的用户数 $NumU_i$。项目 i 的流行度 $PopItem_i=NumU_i$，对其进行归一化，使得项目流行度的取值范围保持在[0,1]之间。归一化公式如式（5）所示.

$$NormPopItem_i=\frac{PopItem_i-MinPop}{MaxPop-MinPop} \tag{5}$$

其中：$NormPopItem_i$ 是归一化后项目 i 的流行度；$PopItem_i$ 是项目 i 的流行度；MinPop、MaxPop 分别是项目流行度的最小值和最大值。

表 1 中展示的评分矩阵包括四个项目，分别统计其项目流行度。$PopItem_1$、$PopItem_2$、$PopItem_3$、$PopItem_4$ 分别是 2、1、0、2。按照式(5)对其进行归一化，则归一化后的项目流行度分别为 1、0.5、0、1。

2.2　流行度差异

流行度差异越小的项目一起出现的可能性越大，本文将

项目间的流行度差异定义为归一化后项目流行度之差的绝对值。当项目间的流行度差异越小，这两个项目由于流行度相近而同时出现的可能性越大，同时被评价的可能性也越大；反之，如果项目间的流行度差异越大，则同时被评价的可能性越小。$NormPopItem_i$ 表示归一化后项目 i 的流行度，$NormPopItem_j$ 表示归一化后项目 j 的流行度，则项目 i 和 j 的流行度差异定义如式（6）所示。

$$PopBias_{i,j} = |NormPopItem_i - NormPopItem_j| \tag{6}$$

按照式(6)计算表 1 中的项目流行度差异，构建项目流行度差异矩阵如表 2 所示。由表 2 可知，$PopBias_{1,2} < PopBias_{1,3}$，因此项目 1 和 2 一起出现的可能性大于项目 1 和 3 一起出现的可能性。

表 2 项目流行度差异矩阵

Table 2 Matrix of item popularity difference

$PopBias_{i,j}$	i_1	i_2	i_3	i_4
i_1	0	0.5	1	0
i_2	0.5	0	0.5	0.5
i_3	1	0.5	0	1
i_4	0	0.5	1	0

2.3 项目流行度和流行度差异对相似度的影响

传统的项目相似度计算方法忽略了项目流行度的影响，事实上流行度高的项目更容易被用户选择和评价，导致大多数流行项目间的相似度偏高。在计算两个项目的相似度时，使用项目流行度阈值将项目分为两个部分，当流行度大于该阈值，则对其相关度进行惩罚。流行度越高的项目，对相似度的贡献越小；反之则对相似度贡献越大，因此项目流行度与惩罚权重正相关。除了项目流行度外，项目间的流行度差异对其相似度也有影响。两个项目间的流行度差异越小，被同时评价的可能性越大，相似度惩罚权重也就越大，流行度差异与惩罚权重负相关。

结合以上分析，根据两个项目的流行度不同本文在相似度计算公式中分别引入惩罚权重式(7)和(8)。

$$w_i = \begin{cases} 1 & NormPopItem_i < \alpha \\ \dfrac{NormPopItem_i}{PopBias_{i,j}} & NormPopItem_i \ge \alpha \end{cases} \tag{7}$$

$$w_j = \begin{cases} 1 & NormPopItem_j < \alpha \\ \dfrac{NormPopItem_j}{PopBias_{i,j}} & NormPopItem_j \ge \alpha \end{cases} \tag{8}$$

其中：w_i 和 w_j 分别是项目 i 和 j 的惩罚函数，反映了流行度不同的两个项目在计算其相似度时体现的不同权重。其中 α 是设定的流行度阈值，如果项目 i 或 j 的流行度小于该阈值，则认为该项目的流行度较低，惩罚权重设置为 1；若是大于该阈值，则认为该项目流行度较高。惩罚权重设置为该项目的流行度与流行度差异的比值，项目的流行度越高，项目间流行度差异越小，则惩罚权重越大。将设计的权重函数引入到相似度计算公式中，改进后的项目相似度计算公式为

$$sim(i,j) = \frac{\sum_{u \in U_i \cap U_j} \left((r_{ui} - \bar{r}_i) w_i\right)\left((r_{uj} - \bar{r}_j) w_j\right)}{\sqrt{\sum_{u \in U_i} \left((r_{ui} - \bar{r}_i) w_i\right)^2} \sqrt{\sum_{u \in U_j} \left((r_{uj} - \bar{r}_j) w_j\right)^2}} \tag{9}$$

其中：$u \in U_i \cap U_j$ 表示两个项目均有评分的用户集合，$u \in U_i$ 表示对项目 i 评分的用户集合，$u \in U_j$ 表示对项目 j 评分的用户集合。

2.4 算法描述

基于以上分析，提出了利用项目流行度对相似度进行惩罚的协同过滤算法。首先根据用户项目评分矩阵 R 计算项目流行度以及项目流行度差异，设置项目流行度阈值 α 分别计算两个项目的惩罚权重，在计算相似度时，利用惩罚权重调节不同流行度的项目对相似度的贡献。算法的具体描述如下：

算法：结合项目流行度加权的协同过滤推荐算法

输入：用户项目评分矩阵 R、目标用户 u、目标项目 i。

输出：目标用户 u 对目标项目 i 的预测评分 P_{ui}。

a)计算项目 i 的流行度 $PopItem_i$，根据式(5)对其归一化；

b)根据定义 6 计算项目流行度差异 $PopBias_{i,j}$；

c)设置流行度阈值 α；

d)利用式(7)和(8)计算项目的惩罚权重 w_i 和 w_j；

e)根据式(9)计算项目间的相似度 $sim(i,j)$，构建项目相似度矩阵 $M_{item}(i,j)$；

f)根据 $M_{item}(i,j)$ 为项目 $i \in I$，找到相似度最大的 K 个项目集合作为项目的最近邻 N_i；

g)利用式(1)计算目标用户 u 对目标项目 i 的评分预测值 P_{ui}。

3 实验结果及分析

本章首先介绍实验所用到的数据集；然后说明评价标准以及对比算法和参数设置；最后给出本文提出的算法与其他算法的对比实验结果，并对实验结果进行分析。

3.1 实验数据集描述

本文实验使用两个数据集，分别是明尼苏达大学 GroupLens 研究组收集的 ML-1M 电影数据集[15]和 Epinion 数据集。在 ML-1M 数据集中，每个用户至少对 20 部电影评分，而在 Epinion 数据集中每个项目至少被评分 1 次。使用 5 折交叉验证对数据集进行划分，从中抽取 80%作为训练集构建模型，20%作为测试集验证算法的效果。

表 3 实验数据集描述

Table 3 Description of datasets used in experiments

数据集名称	用户数(m)	项目数(n)	评分数(r)	稀疏度(s)
ML-1M	6040	3952	1000209	95.81%
Epinion	1071	6131	71833	98.91%

实验数据集描述如表 3 所示。其中稀疏度表示数据集中未评分项目的占比。稀疏度 s 越大，表示该数据集越稀疏，计算方法如式(10)所示。

$$s = 1 - \frac{r \times 100\%}{m \times n} \tag{10}$$

3.2 度量标准

平均绝对误差(mean absolute error，MAE)是推荐系统中常用的推荐质量度量方法，通过预测评分与实际评分的差异计算预测的准确性。MAE 的定义如下：

$$MAE = \frac{\sum_{u,i \in T} |r_{ui} - P_{ui}|}{|T|} \tag{11}$$

其中：T 是测试集；r_{ui} 是用户 u 对项目 i 的真实评分值；P_{ui} 是用户 u 对项目 i 的预测评分值。MAE 值越小，表示预测准确度越高。

覆盖率(coverage)衡量推荐系统中推荐的物品占总物品集合的比例，能有效反映推荐的多样性和新颖性[16]。覆盖率的计算公式为

$$\text{coverage} = \frac{\left|\bigcup_{u \in U} R(u)\right|}{|I|} \tag{12}$$

其中：U 为进行推荐的用户集合；I 为系统中的项目集合；$R(u)$ 是为用户 u 推荐的项目列表。覆盖率较高时，多样性和新颖性也相对较高。

3.3 对比算法及参数设置

为了证明本文算法(PW-IBCF)的有效性，将本文算法与现有的结合项目流行度的推荐算法进行对比实验。对比算法包括基于均值标准化的余弦相似度的项目最近邻方法(NCOS-IBCF) [8]、结合流行度权重的用户协同过滤推荐算法(W-UBCF) [12]、结合项目流行度控制的增强协同过滤算法(ECF) [13]和考虑用户活跃度和项目流行度的基于项目最近邻的协同过滤算法(UA-IBCF) [14]，所有的算法均采取同样的评分预测方法。

本文实验环境为 Windows 10 64 位操作系统，8 GB 内存，Intel(R) Core(TM) i5-4670 CPU @ 3.40 GHz 3.40 GHz，实验代码在 MATLAB R2016a 上运行。数据集的稀疏度不同，其项目的流行度分布也不同。如果数据集的稀疏度 s 较高，其项目流行度则普遍偏低。通过分析实验数据集的项目流行度，本文算法将 ML-1M 数据集的流行度阈值 α 设置为[0.2,0.6]，步长为 0.1；Epinion 数据集的流行度阈值 α 设置为[0.1,0.3]，步长为 0.05；邻居数量取值为[10,100]，步长为 10；推荐列表长度 L=10。

3.4 实验结果及分析

3.4.1 ML-1M 数据集

为了验证不同的流行度阈值 α 对实验结果的影响，通过比较不同邻居数量下的 MAE 值做了多组实验。在 ML-1M 数据集中，随着邻居数量的变化不同的流行度阈值对 MAE 值的影响如图 1 所示。随着流行度阈值 α 的变化，本文算法的 MAE 值保持在[0.76,0.766]间。当流行度阈值 α 取 0.4 时，MAE 值较低且稳定性较好。即当项目流行度大于 0.4 时，对其项目间的相似度进行惩罚，有利于提高评分预测准确度，该数据集的后续实验参数依此设置。

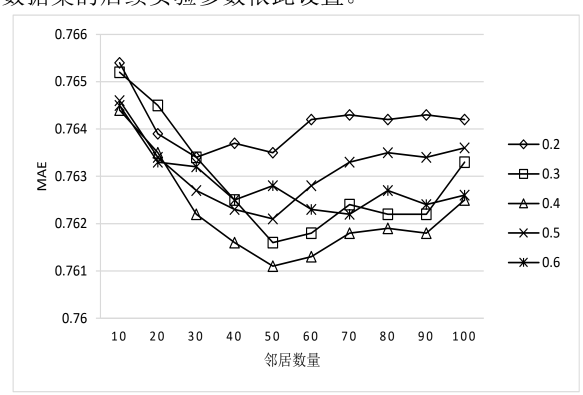


图 1 参数 α 对 MAE 值的影响(ML-1M 数据集)

Fig. 1 Effect of parameter α on MAE (ML-1M)

在 ML-1M 数据集中，对比算法的预测准确度 MAE 的对比结果如图 2 所示。由图 2 可知，NOCS-IBCF 方法的预测准确度较低，可能是因为在惩罚项目间的相似度时并未考虑项目的流行度。W-UBCF、ECF 和 UA-IBCF 方法考虑到项目流行度对相似度的影响，一定程度上提高了预测准确度。在邻居数量为 50 时，本文算法的 MAE 达到最低，约为 0.76，优于其他对比算法，具有更高的预测准确度。

覆盖率的对比实验结果如图 3 所示。基于 NCOS-IBCF 的算法随着邻居数量的增加，覆盖率逐渐降低，说明邻居数量的增加会导致推荐越来越趋向于热门，导致覆盖率下降。而 W-UBCF、ECF、UA-IBCF 以及本文算法 PW-IBCF 随着邻居数量的增加，覆盖率逐渐升高，表明推荐结果的多样性和新颖性也逐渐增加。即考虑项目流行度的推荐算法可以降低热门项目的推荐频率，增加冷门项目的推荐。在邻居数量超过 30 时，本文算法的覆盖率达 30%，高于其他对比算法，且处于上升趋势，表现出更好的推荐性能。

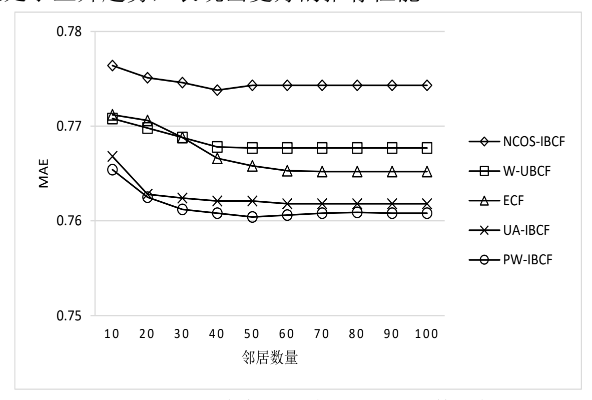


图 2 预测准确度对比结果(ML-1M 数据集)

Fig. 2 Accuracy with different neighbors (ML-1M)

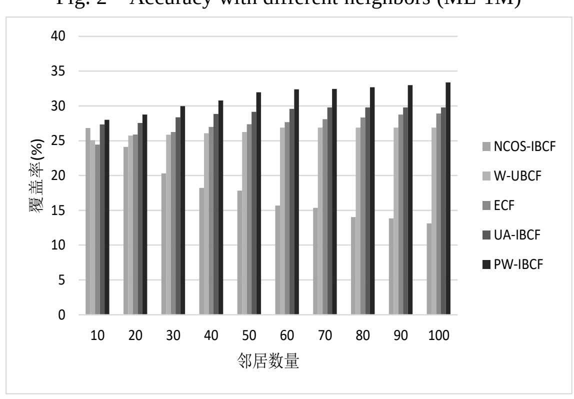


图 3 覆盖率对比结果(ML-1M 数据集)

Fig. 3 Coverage with different neighbors (ML-1M)

3.4.2 Epinion 数据集

在 Epinion 数据集中，不同的流行度阈值对 MAE 值的影响如图 4 所示。随着流行度阈值 α 的变化，本文算法的 MAE 值保持在[0.79,0.8]间。当流行度阈值 α 取 0.15 时，MAE 值较低且稳定性较好。当项目流行度大于 0.15 时，对其项目间的相似度进行惩罚，有利于提高预测准确度，该数据集的后续实验参数 α 依此设置。

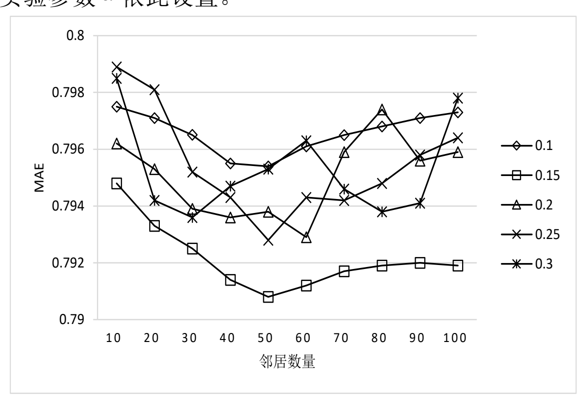


图 4 参数 α 对 MAE 值的影响(Epinion 数据集)

Fig. 4 Effect of parameter α on MAE (Epinion)

在 Epinion 数据集中，对比算法的预测准确度 MAE 的对比结果如图 5 所示。随着邻居数量的变化，对比算法的 MAE 值逐渐降低并趋于平缓。在邻居数量为 50 时，本文算法的 MAE 值达到最低，约为 0.79，优于对比算法。因此，本文算法具有更高的预测准确度。

覆盖率的对比实验结果如图 6 所示。基于 NCOS-IBCF

的算法随着邻居数量的增加，覆盖率呈下降趋势。W-UBCF、ECF、UA-IBCF 以及本文算法 PW-IBCF 随着邻居数量的增加，覆盖率逐渐升高。在邻居数量超过 50 时，本文算法的覆盖率接近 25%，而其他对比算法约为 20%，本文算法的覆盖率明显高于其他对比算法，推荐的多样性和新颖性表现更好。

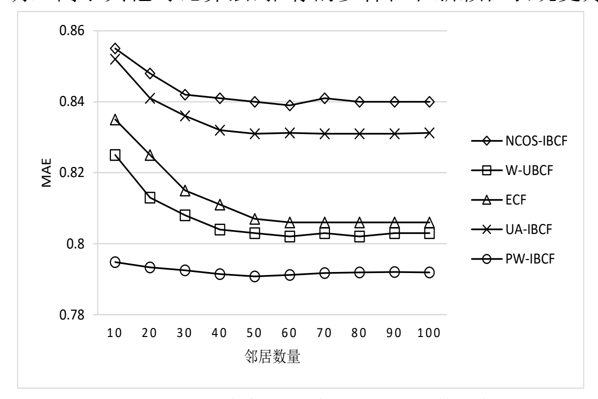


图 5　预测准确度对比结果(Epinion 数据集)

Fig. 5　Accuracy with different neighbors (Epinion)

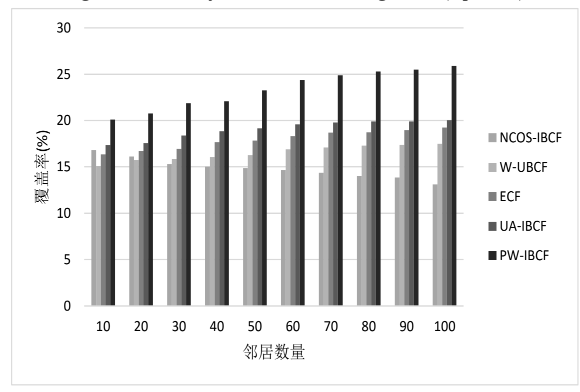


图 6　覆盖率对比结果(Epinion 数据集)

Fig. 6　Coverage with different neighbors (Epinion)

4　结束语

本文改进了利用项目流行度计算项目相似度的度量策略，提出了一种结合项目流行度加权的协同过滤推荐算法。在传统的项目协同过滤算法的基础上，对流行度较高的项目设计相似度惩罚函数，提高相似度计算的准确性，并缓解流行度偏差问题。实验结果表明，当项目流行度超过一定阈值时，对大于该阈值的项目相似度进行相关性惩罚，一定程度上可以提高预测准确度以及推荐多样性。考虑到用户对流行项目的偏好程度对相似度也有影响，将用户对流行项目的偏好程度融入相似度计算，以获得更好的推荐性能是后续工作的主要方向。

参考文献：

[1] Bobadilla J, Ortega F, Hernando A. Recommender systems survey [J]. Knowledge-Based Systems, 2013, 46 (1): 109-132.

[2] Ekstrand M D, Riedl J T, Konstan J A. Collaborative filtering recommender systems [J]. ACM Trans on Information Systems, 2011, 4 (2): 81-173.

[3] Wang Jun, Vries D, *et al.* Unifying user-based and item-based collaborative filtering approaches by similarity fusion [C]// Proc of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. New York: ACM Press, 2006: 501-508.

[4] Liu Limin, Zhang Pengxiang, Lin Le, *et al.* Research of data sparsity based on collaborative filtering algorithm [J]. Applied Mechanics & Materials, 2014, 462-463: 856-860.

[5] Linden G, Smith B, York J. Amazon. com recommendations: item-to-item collaborative filtering [J]. IEEE Internet Computing, 2003, 7 (1): 76-80.

[6] 周军锋，汤显，郭景峰. 一种优化的协同过滤推荐算法 [J]. 计算机研究与发展，2004, 41 (10): 1842-1847. (Zhou Junfeng, Tang Xian, Guo Jingfeng. Optimized collaborative filtering recommendation algorithm [J]. Journal of Computer Research & Development, 2004, 41 (10): 1842-1847.)

[7] Ahn H J. A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem [J]. Information Sciences, 2008, 178 (1): 37-51.

[8] Ekstrand M D, Ludwig M, Konstan J A, *et al.* Rethinking the recommender research ecosystem: reproducibility, openness, and LensKit [C]// Proc of ACM Conference on Recommender Systems. New York: ACM Press, 2011: 133-140.

[9] Steck H. Item popularity and recommendation accuracy [C]// Proc of ACM Conference on Recommender Systems. New York: ACM Press, 2011: 125-132.

[10] Jannach D, Lerche L, Kamehkhosh I, *et al.* What recommenders recommend: an analysis of recommendation biases and possible countermeasures [J]. User Modeling and User-Adapted Interaction, 2015, 25 (5): 427-491.

[11] Abdollahpouri H, Burke R, Mobasher B. Controlling popularity bias in learning-to-rank recommendation [C]// Proc of the 11th ACM Conference on Recommender Systems. New York: ACM Press, 2017.

[12] Zhao Xiangyu, Niu Zhendong, Chen Wei. Opinion-based collaborative filtering to solve popularity bias in recommender systems [C]// Lecture Notes in Computer Science. Berlin: Springer-Verlag, 2013: 426-433.

[13] Chen Tu, Tian Hui, Zhu Xuzhen. An enhanced collaborative filtering with flexible item popularity control for recommender systems [C]// Proc of International Conference on Social Computing. New York: ACM Press, 2014: 1-6.

[14] 王锦坤，姜元春，孙见山，等. 考虑用户活跃度和项目流行度的基于项目最近邻的协同过滤算法 [J]. 计算机科学，2016, 43 (12): 158-162. (Wang Jinkun, Jiang Yuanchun, Sun Jianshan, *et al.* Item-based collaborative filtering algorithm integrating user activity and item popularity [J]. Computer Science, 2016, 43 (12): 158-162.)

[15] Harper F M, Konstan J A. The MovieLens datasets [J]. ACM Trans on Interactive Intelligent Systems, 2016, 5 (4): 1-19.

[16] Shani G, Gunawardana A. Evaluating recommendation systems [M]// Recommender Systems Handbook. Boston: Springer-Verlag, 2011: 257-297.